Clean Copy of Specification Amendments

Paragraph [0016]:

--Figure 5 illustrates a process used to assign a virtual address to a recipient in accordance with one embodiment of the invention.--

Paragraph [0023]:

--IPv6 uses 16 octets, or 128 bits, for addresses, as compared to 4 octets, or 32 bits, in IPv4. The increased number of octets allows IPv6 to provide trillions of possibilities for addresses. Additionally, IPv6 changes the address notation from periods (.) separating address elements, to colons (:). The header format has been modified in IPv6 in order to reduce overhead of the packet headers. Furthermore, IPv6 includes extensions to allow the protocol to be adapted to specialized information. In view of the present invention, this specialized information is the security context, *i.e.*, Supernet ID, Channel ID, virtual address of the recipient.--

Paragraph [0027]:

--Figure 5 illustrates a process used to assign a virtual address to a recipient in accordance with one embodiment of the invention. The virtual network interface includes a client (106), a Supernet authentication secure daemon (SASD) (108), and a virtual address resolution protocol daemon (VARPD) (110). To exchange addresses using the virtual network interface, the client (106) executes a "Supernet Attach" command (denoted as data transfer 100) to the SASD (108). The Supernet Attach command retrieves the virtual address of the recipient, *i.e.*, IPv6 address of the recipient as well as the corresponding Supernet ID, and Channel ID. The SASD (108) responds by replying (denoted as data transfer 102) to the client (106) with the Supernet configuration information, which includes the IPv6 address containing an encoded security context, *i.e.*, virtual address of the recipient, Supernet ID, and Channel ID. The SASD (108) also registers the mapping of the IPv6 address information (denoted as data transfer 104) with the VARPD (110). The VARPD (110) maps the virtual address of the recipient within the Supernet to an actual IPv4

address on the network. For example, a computer on a network may be assigned an IP address of 63.207.206.001. If the computer is subsequently added to a Supernet, it will be assigned a virtual address, *e.g.*, 10.0.0.100. VARPD (110) subsequently maps the virtual address, *i.e.*, 10.0.0.100, to the actual address, *i.e.*, 63.207.206.001.

Paragraph [0028]:

-- The encoded security context of a 128-bit IPv6 address is a unique, proprietary address, such that the first 16 bits are set to a specific value that denotes a site local IPv6 address, as defined by the IPv6 protocol. The first 16 bits include 12 bits for a prefix at the beginning of the address that always have the value "0xfeb". The next 4 bits each have a value of zero. The remaining 112 bits encode the Supernet ID, Channel ID, and the virtual address of the recipient.--

Please amend paragraph [0029] as follows:

--In one or more embodiments of the present invention, the remaining 112 bits include 64 bits for the Supernet ID, 24 bits for the Channel ID, and 24 bits for the virtual address of the recipient in an IPv4 address format. The combination of the Supernet ID and Channel ID identifies the keying material for the encrypt/decrypt. The following example illustrates the aforementioned embodiment. Consider a recipient with a virtual IPv6 address of 10.0.0.1, a Supernet ID of 0x02, and a Channel ID of 0x03. The resulting 128-bit IPv6 address, in accordance with the aforementioned embodiment, is feb0:0000:0000:0000:0002:0000:0300:0001.--

Please amend paragraph [0031] as follows:

--Figure 6 illustrates an exemplary data flow diagram, in accordance with one embodiment of the invention, of the typical packet make-up at various stages during the transfer of data from a client to a recipient within a Supernet. Initially, data (90) that is to be sent is encrypted using the Channel ID and Supernet ID to produce encrypted data (92). The encrypted data (92) is subsequently prepended with an IPv6 header (80) to produce an IPv6 packet (94). The IPv6 packet is subsequently

prepended with an IPv4 header (58) to produce an IPv4 packet (96). The IPv4 header (58) is used to route the data to a recipient computer. Once the IPv4 packet reaches the recipient computer, the IPv4 header (94) is stripped from the IPv4 packet (96) to produce the IPv6 packet (94). The IPv6 packet (94) is then forwarded to a packet handler, such as Netfilter. Netfilter is a packet managing infrastructure provided by the Linux™ kernel. Those skilled in the art will appreciate that any packet management infrastructure may be used. The packet handler uses the 128-bit address embedded within the IPv6 header (80) to decrypt and authenticate the encrypted data (92). The data (90) is then forwarded to the appropriate process within the recipient computer.--

Paragraph [0032]:

--Figure 7 illustrates an exemplary flow process in accordance with one or more embodiments of the present invention. A client process initiates a request to send a packet to a recipient computer on a Supernet (Step 106). The client process forwards data, *i.e.*, the payload (82 in Figure 4), to a virtual network interface (Step 108). At the virtual network interface, encryption and authentication operations on the data are performed to produce encrypted data (Step 110). The virtual network interface subsequently prepends the encrypted data with an IPv6 header to produce an IPv6 packet (Step 112). The IPv6 packet is subsequently prepended with an IPv4 header (Step 114). The IPv4 packet is then forwarded to the recipient, as specified in the IPv4 header (Step 116).--

Paragraph [0033]:

--The recipient receives the IPv4 packet and strips off the IPv4 header to produce the IPv6 packet (Step 118). A handler mechanism takes the IPv6 packet and decodes the security context, *i.e.*, virtual address, Supernet ID and Channel ID (Step 120). Using the security context, the handler mechanism decrypts and authenticates the data within the payload portion of the packet (Step 122). Finally, the virtual address is used as the DA for routing the packet to a corresponding user process (Step 124).--